Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Currently Amended) A method for ~~processing~~ transmitting Internet Protocol (IP) datagrams from a host system via a hardware offload engine that is used for completely offloading transmission control protocol (TCP)/IP protocol stack processing from the host system, ~~using an outbound processing state machine in an outbound processor, wherein the IP datagrams are generated by host system,~~ comprising:

(a) creating an input/output control block ("IOCB") with a plurality of host memory addresses for a memory location within a host memory storing ~~that define~~ host data for transmission ~~to be sent~~ and a host memory address of a network control block ("NCB"), the NCB being used [[to]] for building network protocol headers,

(b) sending the IOCB to the hardware offload engine; wherein the host sends the IOCB to [[the]] an outbound processor of the hardware offload engine;

(c) accessing the NCB stored at the host memory; wherein the outbound processor of the hardware offload engine reads the NCB from host memory using the IOCB;

(d) creating an IP header and media access control (MAC) level protocol header for a single IP packet, if a datagram can fit into the single IP packet; wherein the outbound processor, and not the host system, builds the IP and MAC header based on NCB fields;

(e) transmitting the datagram from step (d); wherein the hardware offload engine transmits the datagram as a single IP packet;

(f) if a datagram size is greater than a certain size, then generating a plurality of IP packets for transmitting the datagram, where each IP packet is a fragmented IP datagram; wherein the outbound processor of the hardware offload engine generates the plurality of IP packets, sets an IP packet length field in a last of the plurality of IP packets, such that the IP packet length field in the last IP packet is different from a IP packet length field value in the other plurality of IP packets; and

(g) setting a flag in each fragmented IP datagram for indicating which fragmented IP datagram is transmitted by the hardware offload engine.

2. (Cancelled)

3. (Cancelled)

4. (Cancelled)

5. (Currently Amended) The method of Claim 1 [[4]], wherein if fragmentation or an IP header is not needed, then the host system sets a descriptor value in the IOCB to indicate to the hardware offload engine that a datagram does not need to be fragmented and an IP header does not need to be created, and the hardware offload engine sends the datagram without fragmenting and without creating an IP header. ~~fragment flags indicate which particular fragment is being sent at any given time~~.

6. (Currently Amended) A method for transmitting transmission control protocol (TCP) packets from a host system via a hardware offload engine that is used for offloading transmission control protocol (TCP)/IP protocol stack processing from the host system ~~processing TCP data packets generated by a host system using an outbound processing state machine in an outbound processor~~, comprising:

(a) creating an input/output control block ("IOCB") with a plurality of host memory addresses for a memory location within a host memory storing ~~that define~~ host outbound TCP data for transmission, ~~to be sent~~ and a host memory address of a network control block ("NCB"), the NCB being used [[to]] for building network protocol headers,

(b) sending the IOCB to the hardware offload engine; wherein the host sends the IOCB to [[the]] an outbound processor of the hardware offload engine;

(c) accessing the NCB stored at the host memory; wherein the outbound processor of the hardware offload engine reads the NCB from the host memory to a local memory for the hardware offload engine;

(d) verifying if a TCP window is open; wherein the outbound processor verifies if the TCP window is open;

(e) building a TCP header from the NCB stored at the local memory; wherein the outbound processor builds the TCP header which includes setting a source port number, a destination port number, a TCP sequence number, a flag to indicate a type of packet, a field whose value shows a TCP header length, and a field whose value indicates a TCP window size;

(f) sending the outbound TCP data to an outbound Internet Protocol processor (OIP) of the hardware offload engine for processing; where while the outbound TCP data is being sent, the outbound processor determines a TCP checksum for the outbound TCP data;

~~building TCP/IP/MAC headers;~~ and

(g) sending the outbound TCP data ~~packet(s)~~.

7. (Cancelled)

8. (Cancelled)

9. (Cancelled)

10. (Cancelled)

11. (Currently Amended) The method of Claim 6, further comprising:

linking the NCB to a re-transmission timer list:

updating the NCB with a last sequence number of the transmitted outbound TCP data ~~transmitted~~; and

linking an original IOCB to the NCB, as a delayed request, in case all the outbound TCP data was not transmitted after a TCP window for a connection transferring the outbound TCP data is closed. ~~due to a window closing or if a re-transmission is necessary; and~~

~~storing the NCB waiting for an Acknowledgement~~.

12. (Currently Amended) A method for processing a TCP data transmit request after a closed TCP window is ~~closed and then~~ reopened based on receiving ~~by the reception of~~ an acknowledgement (ACK) packet ~~using an outbound processor~~, comprising:

reading a network control block (NCB) into a local memory ; wherein a host system creates the NCB at a host system memory after a TCP connection is established, and the NCB is copied to a local memory at a hardware offload engine that is used for completely offloading TCP/Internet Protocol (IP) protocol stack execution from the host system; and wherein an outbound processor in the hardware offload engine copies the NCB from the host system memory to the local memory;

reading a delayed request input/output control block (IOCB) ~~(IOCB)~~ linked to the NCB; wherein the delayed request occurs after the TCP connection with the closed TCP window receives an ACK packet;

verifying if [[a]] the closed TCP window is open;

building ~~TCP/IP/MAC~~ headers for transmitting TCP packets; wherein the outbound processor for the hardware offload engine builds the headers using information from the NCB; and

sending data using the outbound processor built headers. ~~the data packet(s).~~

13. (Original) The method of Claim 12, wherein the outbound processor determines if a requested data transfer has been completed and generates an outbound TCP completion message.

14. (Currently Amended) A method for processing ~~fragmented~~ Internet Protocol (IP) datagram[[s]] fragments received from a network, comprising:

receiving [[the]] an IP datagram fragment[[s]] for an IP datagram and storing the IP datagram fragment ~~into buffers~~ in a local memory of a hardware offload engine, where the hardware offload engine is coupled to a host system and the hardware offload engine completely offloads transmission control protocol (TCP)/Internet Protocol (IP) (TCP/IP) protocol stack execution from the host system;

linking the IP datagram fragment to a reassembly list for ~~a particular~~ the IP datagram; and

when all fragments for the IP datagram are received ~~present~~, sending the complete IP datagram to a destination ~~TCP or a host~~ for additional processing.

15. (Currently Amended) The method of Claim 14, wherein a new reassembly list is created, if the IP fragment is [[the]] a first fragment for the IP datagram ~~received for a datagram, a new reassembly list is created~~.

16. (Currently Amended) The method of Claim 15, wherein after [[a]] the reassembly list is created, a timer is started to ensure the reassembly is completed in a certain amount of time.

17. (Currently Amended) The method of Claim 15, wherein if the IP datagram fragment is not the first fragment ~~received for a datagram~~ and is in order with [[the]] respect to

other fragments already on [[the]] an existing reassembly list, then the received IP datagram fragment is added to the end of the existing reassembly list.

18. (Currently Amended) The method of Claim 14, wherein if the IP datagram fragment is not [[the]] a first datagram fragment ~~received for a datagram~~ and is out of order, ~~with the~~ compared to other datagram fragments already on [[the]] an existing reassembly list, then the datagram fragment is inserted into the existing reassembly list as indicated by [[its]] an IP offset value.

19. (New) A hardware engine for offloading transmission control protocol (TCP)/Internet Protocol (IP) processing from a software stack at a host system, comprising:

a TCP Table manager that interfaces with (a) an outbound TCP processor for processing TCP packets sent by the host system, (b) an inbound TCP processor for processing TCP packets received from a network device, (c) an inbound processor for processing non-TCP network packets received from a network device, and (d) an outbound processor for processing non-TCP packets sent by the host system, where the TCP Table Manager maintains timer functions for all TCP connections at any given time in the hardware engine and maintains a linked list of network control blocks for processing network data sent by the hardware engine and received by the hardware engine, wherein the hardware engine completely offloads execution of a TCP/IP protocol stack in hardware from a host system that executes the TCP/IP protocol stack in software.

20. (New) The hardware engine of Claim 19, wherein the outbound TCP processor requests the TCP Table Manager to read a network control block for an existing TCP connection stored at local memory of the hardware engine and place the network control block in a register set maintained by the TCP Table Manager and accessible by the outbound TCP processor.

21. (New) The hardware engine of Claim 19, wherein the outbound TCP processor requests the TCP Table Manager to read a network control block for a TCP connection from a host system memory and links the network control block to a hash value generated by a hash table maintained by the TCP Table manager.

22. (New) The hardware engine of Claim 19, wherein the TCP Table Manager includes a command processor that arbitrates between a plurality of command sources and translates a received command to an output action(s) to other TCP Table Manager components.

23. (New) The hardware engine of Claim 22, wherein the TCP table manager's command processor co-ordinates inbound and outbound channel access to network control blocks.

24. (New) The hardware engine of Claim 19, wherein the TCP Table Manager provides a read / write register interface to allow simultaneous access to fields within the network control blocks.

25. (New) The hardware engine of Claim 24, wherein the TCP table manager's register interface provides a locking mechanism to allow sole access to a particular field within a network control block.

26. (New) The hardware engine of Claim 19, wherein the TCP table manager includes a timer list manager that maintains timer functions for all TCP connections that are handled by the hardware engine.

27. (New) The hardware engine of Claim 26, wherein the timer list manager maintains a persist timer, an idle timer, a delayed acknowledgement (ACK) timer and a retransmit timer for each TCP connection.

28. (New) The hardware engine of Claim 26, wherein the timer list manager scans a timer list and checks for timed events that have expired at any given time.

29. (New) The hardware engine of Claim 19, wherein the TCP table manager maintains an outbound request list to signal if a network control block is ready for transmit processing by the outbound TCP processor due to a timer event.